

SYSTEM AND METHOD FOR GROUPING DIVERSE OPERATIONS

BACKGROUND OF THE INVENTION

5 1. Field of the Invention

The present invention relates to software systems that use diverse operations including transactions and messages and, more particularly, to a method and system to group these operations.

10 2. Description of Prior Art

Software engineering for enterprise-scale systems presents some important challenges. These include the need to integrate diverse software components in a distributed, heterogeneous system environment while ensuring properties such as reliability, scalability, evolvability, etc. In recent years, a category of software commonly referred to as middleware has emerged to address these needs. Middleware is connectivity software that defines a standard for different software components to interoperate with each other. Middleware provides basic communication mechanisms between components while hiding any complexities introduced through distribution and heterogeneity of the components. In addition, middleware provides higher-level services to facilitate application software development and to address software quality concerns.

20 Two principle kinds of middleware are object-oriented middleware (OOM) and message-oriented middleware (MOM), both contribute to the development of reliable software systems. OOM, for example, provides support for software reliability through distributed transactions. Alternatively, MOM provides support for reliable asynchronous communication between distributed components. Both OOM and MOM have evolved individually and

independently over the past years, and they have been implemented in the development of many enterprise software systems.

The integration and interoperation of services becomes a concern when using OOM and MOM in the same system. In particular, while transactions and messages are reliable individually, existing middleware does not directly support an equivalent level of reliability for use of OOM and MOM in combination.

OOM can be exemplified by object request brokers (ORB), for example, the Object Management Group's Common Object Request Broker Architecture (CORBA) and component technology (e.g., Sun's Enterprise JavaBeans (EJB)). The software components to be integrated are rendered as distributed objects that offer well-defined interfaces. A synchronous client/server model is the standard communications model between objects.

MOM can be exemplified by message queuing (MQ) systems, for example, IBM's MQSeries and implementations of the Java Message Service (JMS) application program interface. Components integrated by MOM communicate by means of asynchronous message exchange, for example, with message queues as intermediators. Components read or write messages (data) using queues, and the distribution of the messages through the network is the responsibility of the MOM. Thus, the components do not directly interact with each other. Communicating components may be anonymous to each other and are typically decoupled.

Each type of middleware has particular advantages. The object-oriented model of OOM is consistent with the object paradigm commonly followed in modern system development and thus helps to support consistent development processes. OOM further aims to promote extensibility and reusability by strictly separating the interfaces and implementations of a component. On the other hand, MOM is particularly well-suited in cases where a decoupling in

time or space is needed or desired. In addition, a multiplicity of communication partners can be supported through multicast notification using MOM. Consequently, OOM and MOM are often used in combination. For example, message queuing may be used to communicate with legacy backend servers, while distributed object calls may be used to communicate with web frontends.

5 One important approach to software reliability is the use of transactions. OOM object transactions resemble database transactions: a transaction transforms a shared state of a system from one consistent state to another consistent state based on an all-or-nothing execution model. The actions that constitute a transaction are synchronous requests on objects. Support for transaction processing has been successfully introduced with services like the CORBA Object Transaction Service (OTS), and its Java binding JTS and the Java Transaction API (JTA).

MOM also has a notion of transaction. However, MOM transactions are different from OOM transactions. A transaction in MOM refers to the grouping of a set of produced or consumed messages as one atomic unit of work: either all messages that constitute a MOM transaction are sent out and read, or none of the messages are sent out and read.

Referring to Fig. 1, a software system using transactions and messages is shown. The system includes a component 110 including both a transactional client and a messaging component. The system also includes a transactional resource 120 and a messaging component 130. Component 110 performs synchronous invocations on the transactional resource 120 within a transaction context. Independent of this transaction context, component 110 communicates asynchronously with the messaging component 130. The system depicted in Fig. 1 is a system illustrating a software system architecture common to existing transaction and messaging systems. A transaction involving one or more transactional resources is independent of any message exchanged between one or more messaging components. Component 110 uses

transactions and messaging in an uncoupled way. Therefore, no properties are observed for the operations as a group.

Existing approaches to grouping operations into larger units to ensure certain properties are limited in that support is only provided for grouping of operations of a homogeneous kind.

- 5 For example, database or object transactions only support the grouping of synchronous requests on transactional resources. Alternatively, messaging units of work only support the grouping of messages. Thus, operations of a heterogeneous kind are not dependent on each other in a group.

There is no known support for grouping diverse operations. As a consequence, software architects need to separate transactions and messages. Therefore a need exists for a system and method for integrating distribution object transaction processing and asynchronous messaging by grouping diverse operations.

SUMMARY OF THE INVENTION

According to one embodiment of the present invention a method is provided for grouping at least two diverse operations. The method includes the step of initiating a context grouping the operations, wherein the group is one of, at least two messaging operations, and one or more messaging operations and a transactional operation. The method further includes the steps of performing the operations within the context, each operation resulting in an outcome, combining the outcomes, determining an overall outcome based on a combination of the outcomes for each operation, and taking at least one action dependent on the overall outcome. The method also includes the step of terminating the context upon taking the action. Each operation is supported by an object.

The outcome of each messaging operation is independent of other messaging operation outcomes. The outcome of a messaging operation is independent of a transactional operation outcome. An operation is one of a synchronous invocation on a transactional resource and a conditional asynchronous message between two or more messaging components. The synchronous invocation occurs in at least one transaction. The asynchronous message results in an outcome, the outcome defined by a condition associated to a corresponding operation.

The method includes the step of grouping the synchronous invocation in the transaction and the conditional asynchronous message. The method further includes the step of interpreting each outcome as one of a success and a failure. The method includes the step of interpreting the overall group outcome as one of a success and a failure. The method evaluates the overall group outcome as a failure if at least one individual operation is interpreted as a failure.

The action is one of a predefined action, an automatically invoked action, and a performed action. The action taken upon determining the overall outcome to be a failure further includes the step of undoing an operation. The action taken upon determining the overall outcome to be a failure further includes the step of compensating for an operation.

According to an embodiment of the present invention, a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform method steps for grouping at least two diverse operations within an object-oriented software system.

The method including the step of initiating a group context by creating a representation of the context according to a defined data structure, and filling the representation with values that identify the group context.

According to one embodiment of the present invention, a method is provided for managing a group of two or more operations within a software system. The method includes the steps of initiating a group context including a sub-context for each operation in the group context, wherein each operation is supported by one of a transactional resource and a messaging component, performing the operations within the sub-contexts, each operation resulting in an outcome, coupling the outcomes within the group context, determining an overall outcome of the group context, and taking at least one action dependent on the overall outcome.

The method further includes the step of terminating the group context upon taking one or more actions. The step of initiating the group context includes the steps of creating a representation of the context according to a defined data structure, and filling the representation with values that identify the group context.

An operation is one of a synchronous invocation on a transactional resource, an asynchronous message between at least two messaging components, and a second group of at least two operations. The synchronous invocation occurs in at least one transaction. The asynchronous message results in an outcome, the outcome defined by a condition associated to a corresponding operation.

The method further includes the step of grouping the synchronous invocation on the transactional resource and a conditional asynchronous messaging. The method interprets each outcome as one of a success and a failure. The method further includes the step of interpreting the overall group outcome as one of a success and a failure. The method includes evaluating the overall group outcome as a failure if at least one operation is interpreted as a failure. The action can be one of a commit, a rollback, and a compensation. The action can be one of an update, a delete, a make-table, and an append. The action taken upon determining the overall outcome to

be a failure further includes the step of undoing an operation. The action taken upon determining the overall outcome to be a failure further includes the step of compensating for an operation.

Managing the group includes one of achieving a defined property of the software system and preserving a defined property of the software system. The outcome of each messaging operation is independent of other messaging operation outcomes. The outcome of a messaging operation is independent of a transactional operation outcome.

BRIEF DESCRIPTION OF THE DRAWINGS

Preferred embodiments of the present invention will be described below in more detail, with reference to the accompanying drawings:

Fig. 1 illustrates a software system that uses transactions and messages as two different kinds of operations in an uncoupled way;

Fig. 2 illustrates a software system where synchronous transactional invocations and asynchronous messages are performed dependently according to the present invention;

Fig. 3 is a block diagram showing four different component interaction styles of messaging;

Fig. 4 is a block diagram showing the combination of individual outcomes of messaging operations and synchronous transactional operations can be combined to yield an overall result;

Fig. 5 is a block diagram showing a method for achieving or preserving defined properties of a group of operations;

Fig. 6 is a block diagram showing a variety of processes for executing an individual operation and determining the outcome of that operation;

Fig. 7 is a block diagram showing the process of invoking a synchronous request as part of a transaction and determining the outcome of that request, as shown in Fig. 6;

Fig. 8 is a block diagram showing the process of sending a message and determining the outcome of that message send, as shown in Fig. 6;

5 Fig. 9 is a block diagram showing the process of receiving a message and determining the outcome of that message receipt, as shown in Fig. 6;

Fig. 10 is a block diagram showing the process of making a request, receiving a reply, and determining the outcome of that request/reply operation, as shown in Fig. 6;

10 Fig. 11 is a block diagram showing the process of receiving a request, making a reply, and determining the outcome of that request/reply operation, as shown in Fig. 6; and

Fig. 12 is a block diagram showing a software system for coupling transactions and messages according to an embodiment of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

15 The present invention relates to a system and method of managing at least two diverse operations to achieve and/or preserve a defined property. Referring to Fig. 2, the system includes a component 210 that is both a transactional client and a first messaging component. The system also includes a transactional resource 220 and a second messaging component 230. The system performs transactions and messaging in a coupled way within a combined transaction and
20 messaging context (the group context) in order to achieve or preserve a property. For example, a transactional workflow object needs to send out notification messages asynchronously (non-blocking) to multiple components using message queuing and also needs the successful delivery of those messages to final recipients reading from the queues. The distributed object

transaction and the notification messages are thus dependent on each other: the transaction will succeed if all notification messages reach final recipients, and if either the transaction or one of the message deliveries fails, appropriate undo-actions will automatically be taken for both.

Therefore, properties for the combined operations can be defined and preserved within the

5 context as shown in Figs. 5 through 12.

It is to be understood that the present invention may be implemented in various forms of hardware, software, firmware, special purpose processors, or a combination thereof. In one embodiment, the present invention may be implemented in software as an application program tangibly embodied on a program storage device. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (CPU), a random access memory (RAM), and input/output (I/O) interface(s). The computer platform also includes an operating system and micro instruction code. The various processes and functions described herein may either be part of the micro instruction code or part of the application program (or a combination thereof) which is executed via the operating system, or part of the operating system itself, or part of the middleware. In addition, various other peripheral devices may be connected to the computer platform such as an additional data storage device and a printing device.

It is to be further understood that, because some of the constituent system components and method steps depicted in the accompanying Figures may be implemented in software, the actual connections between the system components (or the process steps) may differ depending upon the manner in which the present invention is programmed. Given the teachings of the

present invention provided herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

Fig. 3 depicts four different messaging interaction styles. Each style involves two or more messaging components. Each component may be a sender, a recipient, a requestor, and/or a replier. These roles are assigned to components A 310, 330, 350, and 370 of Fig. 3, respectively.

In the first case (1), the component A 310 sends a message to component B 320. Component A 310 is the message sender producing an outgoing message, and component B 320 is the message recipient. This interaction style illustrates a message delivery. Message delivery will be asynchronous from the sender's perspective in that the sender is independent of the availability and response of the recipient. Synchronous message delivery describes an alternative message delivery model. Note that further variants of this interaction style may involve multiple recipients for one message. Also, any of the message recipients, including component B 320, may remain anonymous to the sender. Multiplicity and anonymity with message exchange are supported by the messaging system used for sending messages.

The second case (2) illustrates component A 330 receiving a message from component B 340. Component A 330 is the message recipient of an incoming message, and component B 340 is the message sender. This interaction style illustrates a message receipt. The message receipt may be initiated by component B 340 at the time the message is sent, or, it may be initiated by component A 330 at the time component A 330 decides to receive incoming messages. In the latter case, an intermediary message store holding incoming messages, such as a message queue, may be used.

The third case (3) depicts component A 350 sending an outgoing message to component B 360 and subsequently receiving an incoming message from the same component B 360. This

interaction style illustrates a request/reply model of messaging. This interaction involves two messages: the first message represents the request, the second message represents the reply to the request message. The two messages are correlated in that the reply message references the request message. The reply message may be a simple acknowledgment of receipt, or may
5 contain results of some processing of component B 360 that was triggered as a consequence to the request message. Assuming that the request message is sent asynchronously, component A 350 can continue its processing after sending the request, the component is not blocked waiting for the reply.

In the fourth case (4), the reverse messaging interaction style of (3) is shown. Component
10 A 370 first receives an incoming asynchronous message from component B 380, and may perform some processing as a consequence of this incoming message, and component A 370 later sends an outgoing message to the component B 380.

These four messaging interaction styles may be implemented in a system using transactions and messaging in combination. The transactional client 210 of Fig. 2, for example,
15 may perform any of the messaging interaction styles shown in Fig. 3.

Fig. 4 shows the overall outcome including the various individual outcomes of the diverse operations according to an embodiment of the present invention. The operations are performed within a grouping context and are computed in a software system including a system for grouping transactions and messages. It should be noted that any two messages are
20 considered to be diverse operations. Similarly, messages are diverse from a transaction.

The combined outcome 410 is dependent on the individual outcomes of the different operations performed under an initiated group context. The individual outcomes comprise zero or more transaction outcomes and one or more messaging outcomes. 420 is the outcome of a

synchronous invocation performed under the initiated context (see Fig. 2). 430 is the outcome of a messaging interaction performed under the initiated context. The messaging interaction outcome can be of any of the types shown in 440, 450, 460 and 470 corresponding to messaging styles 1, 2, 3, and 4 of Fig. 3 respectively. The individual outcomes depicted in Fig. 4 are to be understood in the context of a system that uses dependent transactions and messaging.

Fig. 5 shows a method for achieving or preserving defined properties of a group of operations. Properties may be related to , for example, atomicity, consistency, visibility, persistence, compensation, versioning. In addition, these properties can be defined based on, for example, language, system, installation, application.

Atomicity refers to a context that succeeds or fails as a whole. Success therefore, may be defined as the completion of all requests and the delivery of all messages. Thus, if any individual synchronous request or asynchronous message within the context is not successful then the context must fail as a whole.

Another property can include consistency. Object transactions (synchronous requests) transform a system state from one consistent state to another consistent state, as the input to the transaction is guaranteed to be the output of a previously committed transaction. Consistency of system state is application defined data integrity. At the point in time that a transaction commits, the system state conforms to the defined data integrity. According to one embodiment of the present invention, consistency can also refer to messaging. At the point where a transaction commits, the system state conforms to the defined data integrity and may have any number of messages delivered as part of the context.

Still another example of a property is durability. Durability can be achieved within a context by declaring messages to be durable, whereas synchronous object requests are typically

YOR9-2000-0680US1 (8728-452)

13

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

61

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

113

114

115

116

117

118

119

120

121

122

123

124

125

126

127

128

129

130

131

132

133

134

135

136

137

138

139

140

141

142

143

144

145

146

147

148

149

150

151

152

153

154

155

156

157

158

159

160

161

162

163

164

165

166

167

168

169

170

171

172

173

174

175

176

177

178

179

180

181

182

183

184

185

186

187

188

189

190

191

192

193

194

195

196

197

198

199

200

201

202

203

204

205

206

207

208

209

210

211

212

213

214

215

216

217

218

219

220

221

222

223

224

225

226

227

228

229

230

231

232

233

234

235

236

237

238

239

240

241

242

243

244

245

246

247

248

249

250

251

252

253

254

255

256

257

258

259

260

261

262

263

264

265

266

267

268

269

270

271

272

273

274

275

276

277

278

279

280

281

282

283

284

285

286

287

288

289

290

291

292

293

294

295

296

297

298

299

300

301

302

303

304

305

306

307

308

309

310

311

312

313

314

315

316

317

318

319

320

321

322

323

324

325

326

327

328

329

330

331

332

333

334

335

336

337

338

339

340

341

342

343

344

345

346

347

348

349

350

351

352

353

354

355

356

357

358

359

360

361

362

363

364

365

366

367

368

369

370

371

372

373

374

375

376

377

378

379

380

381

382

383

384

385

386

387

388

389

390

391

392

393

394

395

396

397

398

399

400

401

402

403

404

405

406

407

408

409

410

411

412

413

414

415

416

417

418

419

420

421

422

423

424

425

426

427

428

429

430

431

432

433

434

435

436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485

486

487

488

489

490

491

492

493

494

495

496

497

498

499

500

501

502

503

504

505

506

507

508

509

510

511

512

513

514

515

516

517

518

519

520

521

522

523

524

525

526

527

528

529

530

531

532

533

534

535

536

537

538

539

540

541

542

543

544

545

546

547

548

549

550

551

552

553

554

555

556

557

558

559

560

561

562

563

564

565

566

567

568

569

570

571

572

573

574

575

576

577

578

579

580

581

582

583

584

585

586

587

588

589

590

591

592

593

594

595

596

597

598

599

600

601

602

603

604

605

606

607

608

609

610

611

612

613

614

615

616

617

618

619

620

621

622

623

624

625

626

627

628

629

630

631

632

633

634

635

636

637

638

639

640

641

642

643

644

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

664

665

666

667

668

669

670

671

672

673

674

675

676

677

678

679

680

681

682

683

684

685

686

687

688

689

690

691

692

693

694

695

696

697

698

699

700

701

702

703

704

705

706

707

708

709

710

711

712

713

714

715

716

717

718

719

720

721

722

723

724

725

726

727

728

729

730

731

732

733

734

735

736

737

738

739

740

741

742

743

744

745

746

747

748

749

750

751

752

753

754

755

756

757

758

759

760

761

762

763

764

765

766

767

768

769

770

771

772

773

774

775

776

777

778

779

780

781

782

783

784

785

786

787

788

789

790

791

792

793

794

795

796

797

798

799

800

801

802

803

804

805

806

807

808

809

810

811

812

813

814

815

816

817

818

819

820

821

822

823

824

825

826

827

828

829

830

831

832

833

834

835

836

837

838

839

840

841

842

843

844

845

846

847

848

849

850

851

852

853

854

855

856

857

858

859

860

861

862

863

864

865

866

867

868

869

870

871

872

873

874

875

876

877

878

879

880

881

882

883

884

885

886

887

888

889

890

891

892

893

894

895

896

897

898

899

900

901

902

903

904

905

906

907

908

909

910

911

912

913

914

915

916

917

918

919

920

921

922

923

924

925

926

927

928

929

930

931

932

933

934

935

936

937

938

939

940

941

942

943

944

945

946

947

948

949

950

951

952

953

954

955

956

957

958

959

960

961

962

963

964

965

966

967

968

969

970

971

972

973

974

975

976

977

978

979

980

981

982

983

984

985

986

987

988

989

990

991

992

993

994

995

996

997

998

999

1000

1001

1002

1003

1004

1005

1006

1007

1008

1009

1010

1011

1012

1013

1014

1015

1016

1017

1018

1019

1020

1021

1022

1023

1024

1025

1026

1027

1028

1029

1030

1031

1032

1033

1034

1035

1036

1037

1038

1039

1040

1041

1042

1043

1044

1045

1046

1047

1048

1049

1050

1051

1052

1053

1054

1055

1056

1057

1058

1059

1060

1061

1062

1063

1064

1065

1066

1067

1068

1069

1070

1071

1072

1073

1074

1075

1076

1077

1078

1079

1080

1081

1082

1083

1084

1085

1086

1087

1088

1089

1090

1091

1092

1093

1094

1095

1096

1097

1098

1099

1100

1101

1102

1103

1104

1105

1106

1107

1108

1109

1110

1111

1112

1113

1114

1115

1116

1117

1118

1119

1120

1121

1122

1123

1124

1125

1126

1127

1128

1129

1130

1131

1132

1133

1134

1135

1136

1137

1138

1139

1140

1141

1142

1143

1144

1145

1146

1147

1148

1149

1150

1151

1152

1153

1154

1155

1156

1157

1158

1159

1160

1161

1162

1163

1164

1165

1166

1167

1168

1169

1170

1171

1172

1173

1174

1175

1176

1177

1178

1179

1180

1181

1182

1183

1184

1185

1186

1187

1188

1189

1190

1191

1192

1193

1194

1195

1196

1197

1198

1199

1200

1201

1202

1203

1204

1205

1206

1207

1208

1209

1210

1211

1212

1213

1214

1215

1216

1217

1218

1219

1220

1221

1222

1223

1224

1225

1226

1227

1228

1229

1230

1231

1232

1233

1234

1235

1236

1237

1238

1239

1240

1241

1242

1243

1244

1245

1246

1247

1248

1249

1250

1251

1252

1253

1254

1255

1256

1257

1258

1259

1260

1261

1262

1263

1264

1265

1266

1267

1268

1269

1270

1271

1272

1273

1274

1275

1276

1277

1278

1279

1280

1281

1282

1283

1284

1285

1286

1287

1288

1289

1290

1291

1292

1293

1294

1295

1296

1297

1298

1299

1300

1301

1302

1303

1304

1305

1306

1307

1308

1309

1310

1311

1312

1313

1314

1315

1316

1317

1318

1319

1320

1321

1322

1323

1324

1325

1326

1327

1328

1329

1330

1331

1332

1333

1334

1335

1336

1337

1338

1339

1340

1341

1342

1343

1344

1345

1346

1347

1348

1349

1350

1351

1352

1353

1354

1355

1356

1357

1358

1359

1360

1361

1362

1363

1364

1365

1366

1367

1368

1369

1370

1371

1372

1373

1374

1375

1376

1377

1378

1379

1380

1381

1382

1383

1384

1385

1386

1387

1388

1389

1390

1391

1392

1393

1394

1395

1396

1397

1398

1399

1400

1401

1402

1403

1404

1405

1406

1407

1408

1409

1410

1411

1412

1413

1414

1415

1416

1417

1418

1419

1420

1421

1422

1423

1424

1425

1426

1427

1428

1429

1430

1431

1432

1433

1434

1435

1436

1437

1438

1439

1440

1441

1442

1443

1444

1445

1446

1447

1448

1449

1450

1451

1452

1453

1454

1455

1456

1457

1458

1459

1460

1461

1462

1463

1464

1465

1466

1467

1468

1469

1470

1471

1472

1473

1474

1475

1476

1477

1478

1479

1480

1481

1482

1483

1484

1485

1486

1487

1488

1489

1490

1491

1492

1

to the group context. If a transaction context is associated to the group context in block 510, then a new transaction log may be created or an existing log adopted; similarly, if a messaging context is associated to the group context in block 510, then a new messaging log may be created or an existing log adopted. If both transaction and messaging contexts are associated with the group context in block 510, then objects and logs to represent and manage both transaction and messaging contexts may be created or adopted. Representations of contexts, logs, and other information may be used in conducting the group context and the operations that it groups, in evaluating the overall outcome of the context, and in additional operations, dependent on the overall outcome, that serve to achieve or preserve desired properties.

In block 520, operations are performed within the context and individual outcomes associated with each of these operations are determined.

In block 530, the overall outcome is determined by combining the individual outcomes of operations performed within the context. The overall outcome might be, for example, a simple discrete binary outcome, such as “success” or “failure”, or a more complex outcome. The technique used to combine individual outcomes can be predefined, or dynamically selected, and it can also be application specific or application independent.. Block 530 can be executed in parallel with block 520.

In block 540, an action is taken dependent on the overall outcome and the operations performed within the context, such that certain properties associated with the group are preserved or achieved. Such an action may be a simple action or a complex action including multiple simpler actions.

In block 550, the context is terminated, and the grouping of operations that achieves or preserves some defined property or properties is ended.

Fig. 6 depicts a variety of operations that may be performed in block 520 (Fig. 5).

Blocks 610 through 650 are shown in Figs. 7 through 11 respectively, include the performance of individual operations and the evaluation of the respective outcomes of those operations. The operations include synchronous operations on transactional resources (610, shown in Fig. 7) and various kinds of messaging operations (e.g., blocks 620 to 650, shown in Figs. 8 through 11). Any of these blocks may be included or omitted in the execution of block 520. Any included blocks may be performed one or more times, in sequence or in parallel. Block 520 may be terminated based on the individual outcomes of one or more of the described operations as evaluated in blocks 610 to 650, or block 520 may be terminated based on considerations independent of these operations and their outcomes. The termination of block 520 at least implies that no more operations may be initiated within it, and it may entail the termination of any ongoing operations that were previously initiated within it.

Fig. 7 presents an elaboration of block 610 (Fig. 6) in which the operation is a synchronous request that is made as part of a transaction. If such operations are grouped with other operations according to the present invention, then the context of the transaction to which the synchronous requests belong is established as part of the grouping context created in block 510 (Fig. 5). When the transaction context is created, likewise a persistent transaction log must be created or adopted and associated to the transaction in the grouping context. In block 710, a synchronous request is made on a transactional resource. This operation is made within the grouping context established in block 510. In block 720, that operation is logged as part of the current transaction, using the log associated to the transaction in the grouping context. In block 730, the outcome of the individual operation is determined. The outcome can be evaluated as

"success" or "failure", although other simple or more complex outcomes may be considered.

Blocks 720 and 730 can be performed in any order or in parallel.

Fig. 8 presents an elaboration of block 620 (Fig. 6) in which the operation is the asynchronous send of a message. This corresponds to the style of messaging interaction described in the first case of Fig. 3. If such operations are grouped with other operations according to the present invention, then a messaging context is established as part of the grouping context created in block 510 (Fig. 5). A persistent message-sent log is also created or adopted and associated to the messaging context in the grouping context. In block 810 a message is sent asynchronously within the messaging context established with the grouping context. In block 820 the sending of the message is logged in the message-sent log associated to the grouping context. In block 830 the outcome of the message send operation is determined. The outcome may be evaluated as "success" or "failure", although other simple or more complex outcomes may be considered. Blocks 820 and 830 can be performed in any order or in parallel.

Fig. 9 represents an elaboration of block 630 (Fig. 6) in which the operation is the receipt of a message. This corresponds to the style of messaging interaction described in the second case of Fig. 3. In block 910 the process waits to receive a message. Block 910 is terminated when a message is received. As shown in Fig. 8, in this case a messaging context is established for the grouping context, and a persistent message-receipt log is created or adopted and associated to the messaging context. Block 910 may also be terminated when a message has not been received, for example, after the expiration of a timeout. In block 920 a determination is made as to whether a message has been received. If a message has been received, then in block 930 the receipt is logged in the associated message-receipt log. Following block 920, if no message has been received, or following block 930, if a message has been received, in block 940

the outcome of the individual operation is determined. The outcome may be evaluated as "success" or "failure", although other simple or more complex outcomes may be considered. Blocks 930 and 940 can be performed in any order or in parallel.

Fig. 10 represents an elaboration of block 640 (Fig. 6) in which the individual operation includes the successive sending and receipt of a message. In particular, this operation corresponds to the role of a requester in a request/reply pattern of messaging, as described in case 3 of Fig. 3. As shown in Fig. 8, in this case a messaging context is established for the grouping context, and a persistent message-sent log is created and associated to the messaging context. In blocks 1010 and 1020 the process sends and logs a message, respectively. In block 1030 the process then decides whether to wait for an incoming message in reply. If a decision is made to wait, then in block 1040 the process waits for a reply message. This waiting may be terminated when a reply message is received or in response to some other condition, action, or event. In block 1050 a control decision is made based on whether a message was received. If a message was received, then this receipt is logged in block 1060. Following block 1030, if a decision has been made not to wait for a reply message, or following block 1050, if no message has been received, or following block 1060, if a message has been received and logged, in block 1070 the outcome of the individual operation is determined. The outcome may be evaluated as "success" or "failure", although other simple or more complex outcomes may be considered. Blocks 1060 and 1070 can be performed in any order or in parallel.

Fig. 11 represents an elaboration of block 650 (Fig. 6) in which the individual operation includes the successive receipt and sending of a message. In particular, this operation corresponds to the role of a replier in a request/reply pattern of messaging, as described in case 4 of Fig. 3. As shown in Fig. 8, in this case a messaging context is established for the grouping

context, and a persistent message-receipt log is created and associated to the messaging context.

In block 1110, the process waits for a request message. This waiting may be terminated when a message is received or in response to some other condition, action, or event. In block 1120 a

control decision is made based on whether a message was received. If a message was received,

then this receipt is logged in block 1130. Processing based on the received message is conducted

in block 1140. Blocks 1130 and 1140 can be performed in any order or in parallel. Following

this processing, in block 1150, a decision is made as to whether to send a reply message in

response to the received request message. If the decision is to send a reply, then this reply is sent

in block 1160 and logged in the sending is logged in the message log in block 1170. Following

block 1120, if no message has been received, or following block 1150, if no reply message is to

be sent, or following block 1170, if a reply message has been sent and logged, in block 1180 the

outcome of the individual operation is determined. The outcome may be evaluated as "success"

or "failure", although other simple or more complex outcomes may be considered. Blocks 1170

and 1180 can be performed in any order or in parallel.

Fig. 12 depicts a software system using a system according to the present invention for coupling transactions and messages according to a method of the present invention. The software system includes the component 1210, which is a transactional client and messaging component that uses transactions and messages in a coupled manner. The component 1210 uses the system of the present invention 1250 for this purpose. The software system also includes the transactional resource 1220, the message sender 1230, and the message recipient 1240. The system of the present invention 1250 includes the transaction-messaging dependency module 1252, the transaction service 1254, and the messaging service 1256.

Component 1210 uses the transaction-messaging dependency module 1252 to initiate the combined context and to terminate the combined context. It may also use module 1252 to abort the combined context. Further, certain operations that are to be part of the combined context may be initiated and performed using module 1252, too. This includes, for example, conditional
5 messaging operations. Module 1252 may also be used for querying purposes to determine individual operation outcomes and overall group outcomes, and to initiate taking actions depending on operation group outcomes.

Synchronous invocations on transactional resources, such as the one illustrated between component 1210 and resource 1220, may not be required to be performed explicitly through the system 1250. Once transactional client 1210 initiates the combined context using system 1250,
10 module 1252 establishes a transaction context using transaction service 1254, and subsequent synchronous invocations by the transactional client on transactional resources are automatically associated with this transaction context. The transactional client can perform operations on transactional resources directly, for those transactional resources that understand the transaction
15 context established by the transaction service 1254 of system 1250.

Messaging operations are preferably conditional messaging operations in order to define and determine individual messaging outcome. Module 1252 uses the messaging service 1256 for actual reliable message deliveries. Message senders, such as message sender 1230 of the software system, and message recipients, such as message recipient 1240 of the software system,
20 may be decoupled from messaging component 1210 through system 1250. That is, message exchange that is part of the combined context is performed entirely through system 1250, in particular module 1252 which in turn uses the messaging service 1256.

System 1250 implements responsibilities described previously as part of the methods of the present invention with reference to Figs. 7 through 11. Such responsibilities include the logging of the diverse operations in different logs, determining individual operation outcomes, determining the overall operation group outcome, observing and receiving incoming messages, 5 deciding upon the wait for incoming messages (which may include time management and synchronization actions), as well as automatically performing actions in reaction to the operation group outcome to ensure the properties defined for the operation group.

Having described embodiments of a system and method for grouping diverse operations, it is noted that modifications and variations can be made by persons skilled in the art in light of the above teachings. It is therefore to be understood that changes may be made in the particular 10 embodiments of the invention disclosed which are within the scope and spirit of the invention as defined by the appended claims. Having thus described the invention with the details and particularity required by the patent laws, what is claimed and desired protected by Letters Patent is set forth in the appended claims. 15